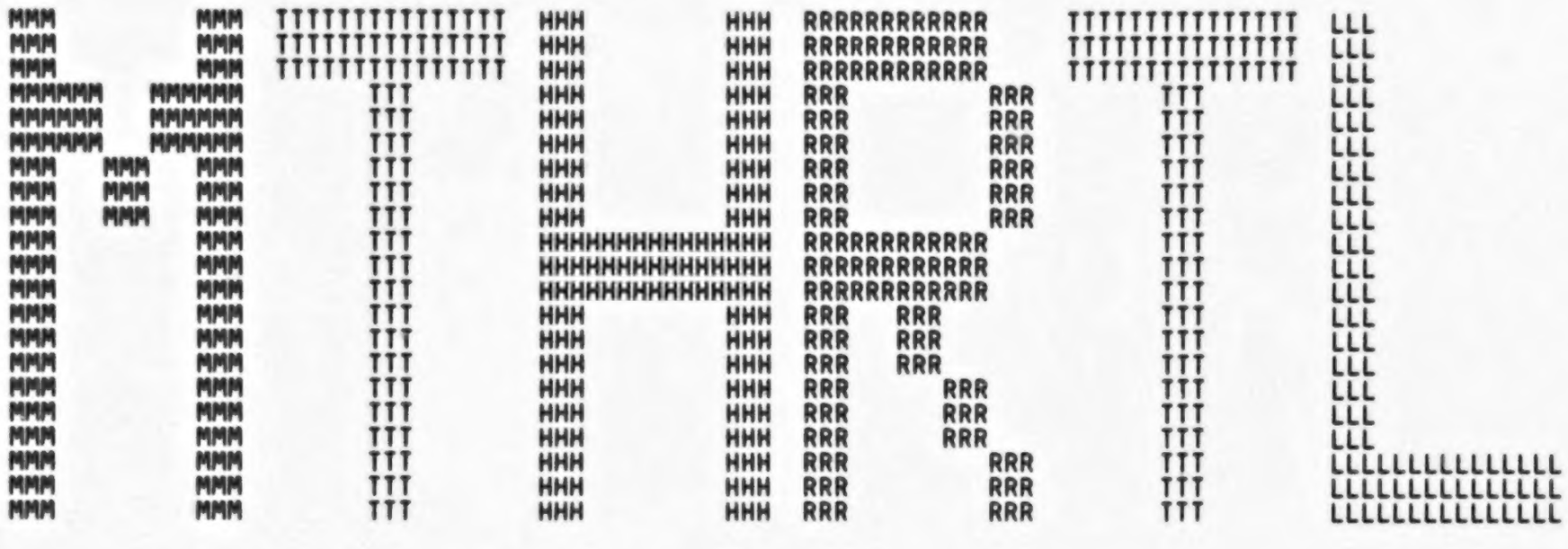
MMM MMM TTTTTTTTTTTTTTT HHH HHH RRRRRRRRRRR TTTTTTTTTTTTTTT LLL
MMMMMM MMMMMM TTT HHH HHH RRR RRR TTT LLL
MMM MMM MMM TTT HHH HHH RRR RRR TTT LLL
MMM MMM MMM TTT HHHHHHHHHHHHHHH RRRRRRRRRRR TTT LLL
MMM MMM MMM TTT HHH HHH RRR RRR TTT LLL
MMM MMM MMM TTT HHH HHH RRR RRR TTT LLL
MMM MMM MMM TTT HHH HHH RRR RRR TTT LLLLLLLLLLLLLLL

```
MM      MM  TTTTTTTTTT  HH      HH  DDDDDDDD     AAAAAA    TTTTTTTTTT    AAAAAA    NN      NN
MM      MM  TTTTTTTTTT  HH      HH  DDDDDDDD     AAAAAA    TTTTTTTTTT    AAAAAA    NN      NN
MMMM  MMMM      TT      HH      HH  DD      DD  AA      AA      TT      AA      AA  NN      NN
MMMM  MMMM      TT      HH      HH  DD      DD  AA      AA      TT      AA      AA  NN      NN
MM  MM  MM      TT      HH      HH  DD      DD  AA      AA      TT      AA      AA  NNNN    NN
MM  MM  MM      TT      HH      HH  DD      DD  AA      AA      TT      AA      AA  NNNN    NN
MM      MM      TT      HHHHHHHHHH  DD      DD  AA      AA      TT      AA      AA  NN  NN  NN
MM      MM      TT      HHHHHHHHHH  DD      DD  AA      AA      TT      AA      AA  NN  NN  NN
MM      MM      TT      HH      HH  DD      DD  AAAAAAAAAA      TT      AAAAAAAAAA  NN    NNNN
MM      MM      TT      HH      HH  DD      DD  AAAAAAAAAA      TT      AAAAAAAAAA  NN    NNNN
MM      MM      TT      HH      HH  DD      DD  AA      AA      TT      AA      AA  NN      NN      ....
MM      MM      TT      HH      HH  DD      DD  AA      AA      TT      AA      AA  NN      NN      ....
MM      MM      TT      HH      HH  DDDDDDDD    AA      AA      TT      AA      AA  NN      NN      ....
MM      MM      TT      HH      HH  DDDDDDDD    AA      AA      TT      AA      AA  NN      NN      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II      /       SS
LL                II      /       SS
LL                II      /       SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  MTH$DATAN       ; Floating Point Arc Tangent Functions
0000     2                                  ; (DATAN,DATAN2,DATAND,DATAND2)
0000     3          .IDENT /2-004/          ; File: MTHDATAN.MAR  EDIT: RNH2004
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28 ;
0000    29 ; FACILITY: MATH LIBRARY
0000    30 ;++
0000    31 ; ABSTRACT:
0000    32 ;
0000    33 ; MTH$DATAN is a function which returns the floating point arctangent
0000    34 ; value (in radians) of its double precision floating point argument.
0000    35 ; MTH$DATAN2 is two argument double floating arctangent. The  call is
0000    36 ; standard call-by-reference.
0000    37 ; MTH$DATAN_R7  is  a  special routine which is the same as MTH$DATAN
0000    38 ; except a  faster non-standard JSB call is used with the argument in
0000    39 ; R0 and no registers are saved.
0000    40 ;
0000    41 ; MTH$DATAND is a function which returns the floating point arctangent
0000    42 ; value (in degrees) of its double precision  floating point argument.
0000    43 ; MTH$DATAND2 is two  argument double floating arctangent. The call is
0000    44 ; standard call-by-reference.
0000    45 ; MTH$DATAND_R7 is  a  special routine which is the same as MTH$DATAND
0000    46 ; except  a  faster non-standard JSB call is used with the argument in
0000    47 ; R0 and no registers are saved.
0000    48 ;
0000    49 ;--
0000    50 ;
0000    51 ; VERSION: 01
0000    52 ;
0000    53 ; HISTORY:
0000    54 ; AUTHOR:
0000    55 ;       Peter Yuo, 15-Oct-76: Version 01
0000    56 ;
0000    57 ; MODIFIED BY:
```

```
0000     58 ;
0000     59 ; 01-1  Peter Yuo, 22-May-77
0000     60 ;
0000     61 ; VERSION: 02
0000     62 ;
0000     63 ; HISTORY:
0000     64 ; AUTHOR:
0000     65 ;       Bob Hanek, 05-Jun-81: Version 02
0000     66 ;
0000     67 ; MODIFIED BY:
0000     68 ;
0000     69 ;
```

```
0000     71
0000     72
0000     73 ; ALGORITHMIC DIFFERENCES FROM FP-11/C ROUTINE:
0000     74 ;       1. To avoid various flags subroutine calls have been used.
0000     75 ;
0000     76 ; Edit History for Version 01 of MTH$DATANDATAN2
0000     77 ;
0000     78
0000     79
0000     80 ; 01-1  Code saving after code review March 1977
0000     81 ;       In DATAN2, fix references to OWN constants so DATAN2 will work.
0000     82 ; 01-3  In MTH$DATAN2, comparison of exponents of arguments X and
0000     83 ;       Y is with 58 instead of 26.
0000     84 ;
0000     85 ; 01-8  - Signal INVALID ARG TO MATH LIBRARY if x=y=0.  TNH 16-June-78
0000     86 ; 01-9  - Fix comments.  TNH 16-June-78
0000     87 ; 01-10 - Move .ENTRY mask to module header.  TNH 14-Aug-78
0000     88 ; 1-011 - Update version number and copyright notice.  JBS 16-NOV-78
0000     89 ; 1-012 - Change MTH_INVARG to MTH$K_INVARGMAT.  JBS 07-DEC-78
0000     90 ; 1-013 - Add "_" to the PSECT directive.  JBS 22-DEC-78
0000     91 ; 1-014 - Declare externals.  SBL 17-May-1979
0000     92 ; 1-015 - Added deree entry points.  RNH 15-MAR-1981
0000     93 ;
0000     94 ;
0000     95 ; Edit History for Version 01 of MTH$DATANDATAN2
0000     96 ;
0000     97 ;
0000     98 ; 2-002 - Use G^ addressing for externals.  SBL 24-Aug-1981
0000     99 ; 2-003 - Changed MTH$DATAND2 entry to MTH$DATAN2D in order to conform
0000    100 ;       to the original specification.  RNH 05-Oct-81
0000    101 ; 2-004 - Un-did previous edit to be consistent with PL/1
0000    102 ;       - Modified small argument processing to avoid a microcode bug
0000    103 ;       in the FPA.  RNH 18-Dec-81
```

```
                        105          .SBTTL  DECLARATIONS    ; Declarative Part of Module
                        106
                        107 ;
                        108 ; INCLUDE FILES:        MTHJACKET.MAR, MTHATAN.MAR
                        109 ;
                        110
                        111 ;
                        112 ; EXTERNAL SYMBOLS:
                        113 ;
                        114          .DSABL  GBL
                        115          .EXTRN  MTH$K_INVARGMAT
                        116          .EXTRN  MTH$$SIGNAL                ; Signal SEVERE error
                        117          .EXTRN  MTH$$AB_ATAN               ; Gobal table used by all Arctangent
                        118                                             ;    routines.  Part of MTHATAN.MAR
                        119 ;
                        120
                        121 ; EQUATED SYMBOLS:
                        122 ;
000040FC                123          ACMASK = ^M<IV, R2, R3, R4, R5, R6, R7> ; .ENTRY register mask, int
                        124                                                  ;  ovf enabled
                        125
                        126 ;
                        127 ; MACROS:       none
                        128 ;
                        129 ; PSECT DECLARATIONS:
                        130
00000000                131          .PSECT  _MTH$CODE       PIC,SHR,LONG,EXE,NOWRT
                        132                                             ; program section for math routines
                        133 ;
                        134 ; OWN STORAGE:  none
                        135 ;
                        136 ; EXTERNALS:
                        137 ;
                        138          .EXTRN  MTH$$SIGNAL                ; Signal a severe error
                        139          .EXTRN  MTH$K_INVARGMAT            ; Invalid argument to math library
                        140          .DSABL  GBL                        ; No other externals allowed
                        141 ;
                        142 ; CONSTANTS:
                        143 ;
```

```
                        0000   145 ;
                        0000   146 ;   ***************************** Constants for DATAN ******************************
                        0000   147 ;
                        0000   148 ;
                        0000   149 ;   Each  entry of the DATAN_TABLE contains the the values of XHI, DATAN_XHI_LO
                        0000   150 ;   and DATAN_XHI_HI respectively.  The table is indexed by a pointer  obtained
                        0000   151 ;   from the MTH$$AB_ATAN table. The MTH$$AB_ATAN table is common to all of the
                        0000   152 ;   arctangent  routines  and is included as part of the MTHATAN module.  NOTE:
                        0000   153 ;   For performance reasons it is important to have  the  DATAN_TABLE  longword
                        0000   154 ;   aligned.
                        0000   155 ;
                        0000   156 ;
                        0000   157          .ALIGN  LONG
                        0000   158
                        0000   159 DATAN_TABLE:
                        0000   160 ; Entry  0
00000000 F87E3ED7       0000   161          .QUAD   ^X00000000F87E3ED7      ;  0.10545442998409271E+00
E21C5BB4 E52DA277       0008   162          .QUAD   ^XE21C5BB4E52DA277      ; -0.83990168661711120E-18
B377B27A 2CE63ED7       0010   163          .QUAD   ^XB377B27A2CE63ED7      ;  0.10506611091781236E+00
                        0018   164 ; Entry  1
00000000 FB703F03       0018   165          .QUAD   ^X00000000FB703F03      ;  0.12888845443725 59E+00
8EC75105 6B81A001       0020   166          .QUAD   ^X8EC751056B81A001      ; -0.27405738718612654E-19
DC7B37BC 422F3F03       0028   167          .QUAD   ^XDC7B37BC422F3F03      ;  0.12818216111847079E+00
                        0030   168 ; Entry  2
00000000 F63E3F1F       0030   169          .QUAD   ^X00000000F63E3F1F      ;  0.15621277689933777E+00
215F05DE B08D22D7       0038   170          .QUAD   ^X215F05DEB08D22D7      ;  0.14615699319155353E-17
3692CB13 ADF03F1E       0040   171          .QUAD   ^X3692CB13ADF03F1E      ;  0.15496040572616338E+00
                        0048   172 ; Entry  3
00000000 E4EC3F47       0048   173          .QUAD   ^X00000000E4EC3F47      ;  0.19520920515060425E+00
7B536972 03519EEA       0050   174          .QUAD   ^X7B536972O3519EEA      ; -0.61942715015325782E-20
CF73AADA 69613F45       0058   175          .QUAD   ^XCF73AADA69613F45      ;  0.19278481107058050E+00
                        0060   176 ; Entry  4
00000000 C3D13F7F       0060   177          .QUAD   ^X00000000C3D13F7F      ;  0.24977041780948639E+00
B4E3255F 9F63A232       0068   178          .QUAD   ^XB4E3255F9F63A232      ; -0.60519693165102660E-18
C6A74C33 A30A3F7A       0070   179          .QUAD   ^XC6A74C33A30A3F7A      ;  0.24476257410146354E+00
                        0078   180 ; Entry  5
00000000 DB973F9F       0078   181          .QUAD   ^X00000000DB973F9F      ;  0.31222221255302429E+00
94AB79A5 57201F4A       0080   182          .QUAD   ^X94AB79A557201F4A      ;  0.10711808370652331E-19
AEB45198 F28D3F9A       0088   183          .QUAD   ^XAEB45198F28D3F9A      ;  0.30263177510309217E+00
                        0090   184 ; Entry  6
00000000 9E8E3FC7       0090   185          .QUAD   ^X000000009E8E3FC7      ;  0.38988155126571655E+00
37BDC0BB D813A29B       0098   186          .QUAD   ^X37BDC0BBD813A29B      ; -0.10560403693868137E-17
D2E26F78 56713FBE       00A0   187          .QUAD   ^XD2E26F7856713FBE      ;  0.37175325856916232E+00
                        00A8   188 ; Entry  7
00000000 33B63FFF       00A8   189          .QUAD   ^X0000000033B63FFF      ;  0.49844139814376831E+00
E8718A35 E17DA331       00B0   190          .QUAD   ^XE8718A35E17DA331      ; -0.24107346684847003E-17
3007B09B BFAF3FEC       00B8   191          .QUAD   ^X3007B09BBFAF3FEC      ;  0.46239995032155439E+00
                        00C0   192 ; Entry  8
00000000 F8EB4026       00C0   193          .QUAD   ^X00000000F8EB4026      ;  0.65223568677902222E+00
05D855B7 DF45A3BB       00C8   194          .QUAD   ^X05D855B7DF45A3BB      ; -0.50922848759855227E-17
B5B6B9DC F4384013       00D0   195          .QUAD   ^XB5B6B9DCF4384013      ;  0.57794527566576090E+00
                        00D8   196 ; Entry  9
00000000 0712405E       00D8   197          .QUAD   ^X000000000712405E      ;  0.86729538440704346E+00
F2BF420E DB20A3C7       00E0   198          .QUAD   ^XF2BF420EDB20A3C7      ; -0.54171066766593593E-17
E63CB34A E62B4036       00E8   199          .QUAD   ^XE63CB34AE62B4036      ;  0.71444962622890612E+00
                        00F0   200 ; Entry 10
00000000 CBD84095       00F0   201          .QUAD   ^X00000000CBD84095      ;  0.11702833175659180E+01
```

```
                                0F8   202          .QUAD   ^X69749B082D47A3EC      ; -0.64015869933736197E-17
69749B08 2D47A3EC               100   203          .QUAD   ^X665302221B62405D      ;  0.86369907905682312E+00
66530222 1B62405D               108   204 ; Entry 11
                                108   205          .QUAD   ^X000000008DEB40D2      ;  0.16449559926986694E+01
00000000 8DEB40D2               110   206          .QUAD   ^XA226B1AA552C242C      ;  0.93421751035229859E-17
A226B1AA 552C242C               118   207          .QUAD   ^X01D0C30925404083      ;  0.10245743706054911E+01
01D0C309 25404083               120   208 ; Entry 12
                                120   209          .QUAD   ^X0000000088054124      ;  0.25708019733428955E+01
00000000 88054124               128   210          .QUAD   ^XD3691BCACD08244B      ;  0.11048069196521280E-16
D3691BCA CD08244B               130   211          .QUAD   ^X45BD59C493CA4099      ;  0.11998227060617363E+01
45BD59C4 93CA4099               138   212 ; Entry 13
                                138   213          .QUAD   ^X000000007D0E41AB      ;  0.53590154647827148E+01
00000000 7D0E41AB               140   214          .QUAD   ^X73B917584359A428      ; -0.91215597452526939E-17
73B91758 4359A428               148   215          .QUAD   ^XDFB5323772D240B1      ;  0.13863165612417540E+01
DFB53237 72D240B1               150   216
                                150   217 ;
                                150   218 ; Tables to be used in POLYD for computing DATAN:  DATANTAB1 is obtained
                                150   219 ; from Hart et. al. (No. 4904).  DATANTAB2 is the same as DATANTAB1 except
                                150   220 ; that C0 is set to 0
                                150   221 ;
                                150   222
                                150   223 DATANTAB1:
                                150   224
4B4F7B0B FCA13E98               150   225          .QUAD   ^X4B4F7B0BFCA13E98      ; C6 = 0.74700604980000000E-01
BA534D4C 1F19BEBA               158   226          .QUAD   ^XBA534D4C1F19BEBA      ; C5 = -.90879628821850000E-01
D5B0D0E5 8E1E3EE3               160   227          .QUAD   ^XD5B0D0E58E1E3EE3      ; C4 = 0.11111091685300320E+00
EEBF86F9 4924BF12               168   228          .QUAD   ^XEEBF86F94924BF12      ; C3 = -.14285714219884826E+00
200FCCC8 CCCC3F4C               170   229          .QUAD   ^X200FCCC8CCCC3F4C      ; C2 = 0.19999999999893708E+00
AA4EAAAA AAAABFAA               178   230          .QUAD   ^XAA4EAAAAAAAABFAA      ; C1 = -.33333333333333269E+00
00000000 00004080               180   231          .QUAD   ^X0000000000004080      ; C0 = 0.10000000000000000E+01
         00000007               188   232 DATANLEN1 = .- DATANTAB1/8
                                188   233
                                188   234 DATANTAB2:
4B4F7B0B FCA13E98               188   235          .QUAD   ^X4B4F7B0BFCA13E98      ; C6 = 0.74700604980000000E-01
BA534D4C 1F19BEBA               190   236          .QUAD   ^XBA534D4C1F19BEBA      ; C5 = -.90879628821850000E-01
D5B0D0E5 8E1E3EE3               198   237          .QUAD   ^XD5B0D0E58E1E3EE3      ; C4 = 0.11111091685300320E+00
EEBF86F9 4924BF12               1A0   238          .QUAD   ^XEEBF86F94924BF12      ; C3 = -.14285714219884826E+00
200FCCC8 CCCC3F4C               1A8   239          .QUAD   ^X200FCCC8CCCC3F4C      ; C2 = 0.19999999999893708E+00
AA4EAAAA AAAABFAA               1B0   240          .QUAD   ^XAA4EAAAAAAAABFAA      ; C1 = -.33333333333333269E+00
00000000 00000000               1B8   241          .QUAD   ^X0000000000000000      ; C0 = 0.00000000000000000E+00
         00000007               1C0   242 DATANLEN2 = .- DATANTAB2/8
                                1C0   243
                                1C0   244 D_PI:
68C2A221 0FDA4149               1C0   245          .QUAD   ^X68C2A2210FDA4149      ; pi
                                1C8   246 D_PI_OVER_2:
68C2A221 0FDA40C9               1C8   247          .QUAD   ^X68C2A2210FDA40C9      ; pi/2
                                1D0   248 D_MPI_OVER_2:
68C2A221 0FDAC0C9               1D0   249          .QUAD   ^X68C2A2210FDAC0C9      ; -pi/2
                                1D8   250 D_PI_OVER_2_HI:
68C2A221 0FDA40C9               1D8   251          .QUAD   ^X68C2A2210FDA40C9      ; High order bits of pi/2
                                1E0   252 D_PI_OVER_2_LO:
03708A2E 131923D3               1E0   253          .QUAD   ^X03708A2E131923D3      ; Low order bits of pi/2
                                1E8   254
```

```
                        01E8   256 ;
                        01E8   257 ; ************************** Constants for ATAND ******************************
                        01E8   258 ;
                        01E8   259 ; Each entry of the DATAND_TABLE contains the the values of XHI, DATAND_XHI_LO
                        01E8   260 ; and DATAND_XHI_HI respectively.  The table is indexed by a pointer  obtained
                        01E8   261 ; from  the MTH$$AB_ATAN table. The MTH$$AB_ATAN table is common to all of the
                        01E8   262 ; arctangent routines and is included as part of the MTHATAN module. NOTE: For
                        01E8   263 ; performance reasons it is important to have the DATAN_TABLE longword aligned.
                        01E8   264 ;
                        01E8   265 ;
                        01E8   266
                        01E8   267 DATAND_TABLE:
                        01E8   268 ; Entry  0
00000000 F87E3ED7       01E8   269                .QUAD   ^X00000000F87E3ED7      ;  0.10545442998409271E+00
EC84E32B 2B2BA44F       01F0   270                .QUAD   ^XEC84E32B2B2BA44F      ; -0.11230634392205251E-16
F76467D8 A29141C0       01F8   271                .QUAD   ^XF76467D8A29141C0      ;  0.60198447254440279E+01
                        0200   272 ; Entry  1
00000000 FB703F03       0200   273                .QUAD   ^X00000000FB703F03      ;  0.12888884544372559E+00
96F9C4C8 A0012420       0208   274                .QUAD   ^X96F9C4C8A0012420      ;  0.87075001607967749E-17
795BCF00 047A41EB       0210   275                .QUAD   ^X795BCF00047A41EB      ;  0.73442968409542958E+01
                        0218   276 ; Entry  2
00000000 F63E3F1F       0218   277                .QUAD   ^X00000000F63E3F1F      ;  0.15621277689933777E+00
FC66745A F63822CA       0220   278                .QUAD   ^XFC66745AF63822CA      ;  0.13753226458048320E-17
5E9101FB 0EA7420E       0228   279                .QUAD   ^X5E9101FB0EA7420E      ;  0.88785772397440363E+01
                        0230   280 ; Entry  3
00000000 E4EC3F47       0230   281                .QUAD   ^X00000000E4EC3F47      ;  0.19520920515060425E+00
728CB36C 241C25C2       0238   282                .QUAD   ^X728CB36C241C25C2      ;  0.84195264883526611E-16
EE5FAC64 BB6A4230       0240   283                .QUAD   ^XEE5FAC64BB6A4230      ;  0.11045756028571212E+02
                        0248   284 ; Entry  4
00000000 C3D13F7F       0248   285                .QUAD   ^X00000000C3D13F7F      ;  0.24977041780948639E+00
72036DE9 3B89A5D5       0250   286                .QUAD   ^X72036DE93B89A5D5      ; -0.92474884414648262E-16
D4B09F5E 61BD4260       0258   287                .QUAD   ^XD4B09F5E61BD4260      ;  0.14023862478771928E+02
                        0260   288 ; Entry  5
00000000 DB973F9F       0260   289                .QUAD   ^X00000000DB973F9F      ;  0.31222221255302429E+00
B8A22FB8 0616A565       0268   290                .QUAD   ^XB8A22FB80616A565      ; -0.49661615106200334E-16
018A1366 B758428A       0270   291                .QUAD   ^X018A1366B758428A      ;  0.17339523459959485E+02
                        0278   292 ; Entry  6
00000000 9E8E3FC7       0278   293                .QUAD   ^X000000009E8E3FC7      ;  0.38988155126571655E+00
344BEAED E7C2A489       0280   294                .QUAD   ^X344BEAEDE7C2A489      ; -0.14951724532714388E-16
F73829B3 662E42AA       0288   295                .QUAD   ^XF73829B3662E42AA      ;  0.21299892736248605E+02
                        0290   296 ; Entry  7
00000000 33B63FFF       0290   297                .QUAD   ^X0000000033B63FFF      ;  0.49844139814376831E+00
752E7920 CC7825F9       0298   298                .QUAD   ^X752E7920CC7825F9      ;  0.10833292304647813E-15
13348584 F2D242D3       02A0   299                .QUAD   ^X13348584F2D242D3      ;  0.26493565600483999E+02
                        02A8   300 ; Entry  8
00000000 F8EB4026       02A8   301                .QUAD   ^X00000000F8EB4026      ;  0.65223568677902222E+00
906EBE73 31EC258D       02B0   302                .QUAD   ^X906EBE7331EC258D      ;  0.61233578397063759E-16
214E9029 748E4304       02B8   303                .QUAD   ^X214E9029748E4304      ;  0.33113825085173017E+02
                        02C0   304 ; Entry  9
00000000 0712405E       02C0   305                .QUAD   ^X000000000712405E      ;  0.86729538440704346E+00
61B2D72E 6F942641       02C8   306                .QUAD   ^X61B2D72E6F942641      ;  0.16777886794863949E-15
A4871377 BD634323       02D0   307                .QUAD   ^XA4871377BD634323      ;  0.40934948257615480E+02
                        02D8   308 ; Entry 10
00000000 CBD84095       02D8   309                .QUAD   ^X00000000CBD84095      ;  0.11702833175659180E+01
F3FAEAF9 FCD82585       02E0   310                .QUAD   ^XF3FAEAF9FCD82585      ;  0.58107895623740531E-16
722AC5D2 F1FB4345       02E8   311                .QUAD   ^X722AC5D2F1FB4345      ;  0.49486311999291994E+02
                        02F0   312 ; Entry 11
```

```
00000000 8DEB40D2  02F0    313          .QUAD   ^X000000008DEB40D2      ;  0.16449559926986694E+01
E4C2D4E7 9E22A6C5  02F8    314          .QUAD   ^XE4C2D4E79E22A6C5      ; -0.34281209639921420E-15
BF6999B3 D0AD436A  0300    315          .QUAD   ^XBF6999B3D0AD436A      ;  0.58703787232967309E+02
                   0308    316 ; Entry 12
00000000 88054124  0308    317          .QUAD   ^X0000000088054124      ;  0.25708019733428955E+01
BEDD635C 359CA65C  0310    318          .QUAD   ^XBEDD635C359CA65C      ; -0.19100122312198548E-15
9ABE70A0 7D534389  0318    319          .QUAD   ^X9ABE70A07D534389      ;  0.68744777221303021E+02
                   0320    320 ; Entry 13
00000000 7D0E41AB  0320    321          .QUAD   ^X000000007D0E41AB      ;  0.53590154647827148E+01
58110A04 52C3A4B5  0328    322          .QUAD   ^X58110A0452C3A4B5      ; -0.19659110337997096E-16
66B57F7F DC34439E  0330    323          .QUAD   ^X66B57F7FDC34439E      ;  0.79430088028242020E+02
                   0338    324
                   0338    325 ;
                   0338    326 ; Tables to be used in POLYD for computing DATAND:  DATANDTAB1 is obtained
                   0338    327 ; by multiplying the coefficients given in Hart et. al. (No. 4904) by
                   0338    328 ; 180/pi.  DATANDTAB2 is the same as DATANDTAB1 except that C0 is set to
                   0338    329 ; 180/pi - 64 instead of 180/pi.
                   0338    330 ;
                   0338    331
                   0338    332 DATANDTAB1:
B22B334C F6004188  0338    333          .QUAD   ^XB22B334CF6004188      ; C6 = 0.42800293924279392E+01
270AAD65 9FE6C1A6  0340    334          .QUAD   ^X270AAD659FE6C1A6      ; C5 = -.52070191752074788E+01
F448F26A B7CC41CB  0348    335          .QUAD   ^XF448F26AB7CC41CB      ; C4 = 0.63661865935060939E+01
404149F1 F637C202  0350    336          .QUAD   ^X404149F1F637C202      ; C3 = -.81851113212942581E+01
DD37DC13 58B34237  0358    337          .QUAD   ^XDD37DC1358B34237      ; C2 = 0.11459155902555563E+02
5F813769 C9EBC298  0360    338          .QUAD   ^X5F813769C9EBC298      ; C1 = -.19098593171027404E+02
0FBED31E 2EE04365  0368    339          .QUAD   ^X0FBED31E2EE04365      ; C0 = 0.57295779513082321E+02
         00000007  0370    340 DATANDLEN1 = .- DATANDTAB1/8
                   0370    341
                   0370    342 DATANDTAB2:
B22B334C F6004188  0370    343          .QUAD   ^XB22B334CF6004188      ; C6 = 0.42800293924279392E+01
270AAD65 9FE6C1A6  0378    344          .QUAD   ^X270AAD659FE6C1A6      ; C5 = -.52070191752074788E+01
F448F26A B7CC41CB  0380    345          .QUAD   ^XF448F26AB7CC41CB      ; C4 = 0.63661865935060939E+01
404149F1 F637C202  0388    346          .QUAD   ^X404149F1F637C202      ; C3 = -.81851113212942581E+01
DD37DC13 58B34237  0390    347          .QUAD   ^XDD37DC1358B34237      ; C2 = 0.11459155902555563E+02
5F813769 C9EBC298  0398    348          .QUAD   ^X5F813769C9EBC298      ; C1 = -.19098593171027404E+02
                   03A0    349 D_PI_OV_180_M_64:
8212670F 88F9C1D6  03A0    350          .QUAD   ^X8212670F88F9C1D6      ; C0 = -.67042204869176791E+01
         00000007  03A8    351 DATANDLEN2 = .- DATANDTAB2/8
                   03A8    352
                   03A8    353
                   03A8    354 D_90:
00000000 000043B4  03A8    355          .QUAD   ^X00000000000043B4      ; 90.
                   03B0    356 D_M90:
00000000 0000C3B4  03B0    357          .QUAD   ^X0000000000000C3B4      ; -90.
                   03B8    358 D_180:
00000000 00004434  03B8    359          .QUAD   ^X0000000000004434      ; 180
                   03C0    360
```

```
                     03C0    362          .SBTTL  MTH$DATAN - Standard Single Precision Floating Arc Tangent
                     03C0    363
                     03C0    364
                     03C0    365 ;++
                     03C0    366 ; FUNCTIONAL DESCRIPTION:
                     03C0    367 ;
                     03C0    368 ; DATAN - double precision floating point function
                     03C0    369 ;
                     03C0    370 ; DATAN is computed using the following steps:
                     03C0    371 ;
                     03C0    372 ;     1. If X > 11 then
                     03C0    373 ;        a. Let W = 1/X.
                     03C0    374 ;        b. Compute DATAN(W) = W*P(W**2), where P is a polynomial of
                     03C0    375 ;           degree 6.
                     03C0    376 ;        c. Set DATAN(X) = pi/2 - DATAN(W)
                     03C0    377 ;     2. If 3/32 =< X =< 11 then
                     03C0    378 ;        a. Obtain XHI by table look-up.
                     03C0    379 ;        b. Compute Z = (X - XHI)/(1 + X*XHI).
                     03C0    380 ;        c. Compute DATAN(Z) = Z*P(Z**2), where P is a polynomial of
                     03C0    381 ;           degree 6.
                     03C0    382 ;        d. Obtain DATAN(XHI) by table look-up.  DATAN(XHI) will have
                     03C0    383 ;           two parts - the high order bits, DATAN_XHI_HI, and the low
                     03C0    384 ;           order bits, DATAN_XHI_LO.
                     03C0    385 ;        e. Compute DATAN(X) = DATAN_XHI_HI + (DATAN_XHI_LO + DATAN(Z)).
                     03C0    386 ;     3. If 0 =< X < 3/32 then
                     03C0    387 ;        a. Compute DATAN(X) = X + X*Q(X**2), where Q is a polynomial
                     03C0    388 ;           of degree 6.
                     03C0    389 ;     4. If X < 0 then
                     03C0    390 ;        a. Compute Y = DATAN(|X|) using steps 1 to 3.
                     03C0    391 ;        b. Set DATAN(X) = -Y.
                     03C0    392 ;
                     03C0    393 ;
                     03C0    394 ; CALLING SEQUENCE:
                     03C0    395 ;
                     03C0    396 ;       Arctangent.wd.v = MTH$DATAN(x.rd.r)
                     03C0    397 ;
                     03C0    398 ; INPUT PARAMETERS:
                     03C0    399 ;
00000004             03C0    400 ;       LONG = 4                        ; define longword multiplier
00000004             03C0    401 ;       x = 1 * LONG                    ; x is an angle in radians
                     03C0    402 ;
                     03C0    403 ; IMPLICIT INPUTS:      none
                     03C0    404 ;
                     03C0    405 ; OUTPUT PARAMETERS:
                     03C0    406 ;
                     03C0    407 ;       VALUE:  double precision floating arctangent angle of the argument
                     03C0    408 ;
                     03C0    409 ; IMPLICIT OUTPUTS:     none
                     03C0    410 ;
                     03C0    411 ; SIDE EFFECTS:
                     03C0    412 ;
                     03C0    413 ; Signals:      none
                     03C0    414 ;
                     03C0    415 ; NOTE: This procedure disables floating point underflow, enable integer
                     03C0    416 ; overflow, causes no floating overflow or other arithmetic traps, and
                     03C0    417 ; preserves enables across the call.
                     03C0    418 ;
```

```
                                03C0   419 ;---
                                03C0   420
                                03C0   421
                         40FC   03C0   422          .ENTRY  MTH$DATAN, ACMASK       ; standard call-by-reference entry
                                03C2   423                                         ; disable DV (and FU), enable IV
                                03C2   424          MTH$FLAG_JACKET                ; flag that this is a jacket procedure
                                03C2
    6D   00000000'GF     9E     03C2               MOVAB   G^MTH$$JACKET_HND, (FP)
                                03C9                                               ; set handler address to jacket
                                03C9                                               ; handler
                                03C9
                                03C9   425                                         ; in case of an error in special JSB
                                03C9   426                                         ; routine
    50    04 BC          70     03C9   427          MOVD    @x(AP), R0             ; R0/R1 = arg
                        6A      10     03CD   428          BSBB    MTH$DATAN_R7           ; call special DATAN rountine
                                04     03CF   429          RET                            ; return - result in R0
                                       03D0   430
```

```
                                03D0   432          .SBTTL  MTH$DATAN2 - Standard Double Floating Arctangent With 2 Arguments
                                03D0   433 ;++
                                03D0   434 ; FUNCTIONAL DESCRIPTION:
                                03D0   435 ;
                                03D0   436 ; DATAN2  - double precision floating point function
                                03D0   437 ;
                                03D0   438 ; DATAN2(X,Y) is computed as following:
                                03D0   439 ;
                                03D0   440 ;       If Y = 0 or X/Y > 2**57, DATAN2(X,Y) = PI/2 * (sign X)
                                03D0   441 ;       If Y > 0 and X/Y =< 2**57, DATAN2(X,Y) = DATAN(X/Y)
                                03D0   442 ;       If Y < 0 and X/Y =< 2**57, DATAN2(X,Y) = PI * (sign X) + DATAN(X/Y)
                                03D0   443 ;
                                03D0   444 ;
                                03D0   445 ; CALLING SEQUENCE:
                                03D0   446 ;
                                03D0   447 ;       Arctangent2.wd.v = MTH$DATAN2(x.rd.r, y.rd.r)
                                03D0   448 ;
                                03D0   449 ; INPUT PARAMETERS:
                                03D0   450 ;
                     00000004   03D0   451 ;       x = 1 * LONG                    ; x is the first argument
                     00000008   03D0   452 ;       y = 2 * LONG                    ; y is the second argument
                                03D0   453 ;
                                03D0   454 ; SIDE EFFECTS: See description of MTH$DATAN
                                03D0   455 ;
                                03D0   456 ;--
                                03D0   457
                                03D0   458
                         40FC   03D0   459          .ENTRY  MTH$DATAN2, ACMASK      ; standard call-by-reference entry
                                03D2   460                                          ; disable DV (and FU), enable IV
                                03D2   461          MTH$FLAG_JACKET                 ; flag that this is a jacket procedure
                                03D2
    6D   00000000'GF    9E      03D2               MOVAB   G^MTH$$JACKET_HND, (FP)
                                03D9                                                ; set handler address to jacket
                                03D9                                                ; handler
                                03D9
                                03D9   462                                          ; in case of an error in special JSB
                                03D9   463                                          ; routine
      50    04 BC    70        03D9   464          MOVD    @x(AP), R0              ; R0/R1 = arg1
      52    08 BC    70        03DD   465          MOVD    @y(AP), R2              ; R2/R3 = arg2
                                03E1   466 ;
                                03E1   467 ; Test if Y = 0 or X/Y > 2**57
                                03E1   468 ;
                   31    13     03E1   469          BEQL    INF                     ; branch to INF if Y = 0
54    50     807F  8F    AB     03E3   470          BICW3   #^X807F, R0, R4         ; R4 = exponent(X)
55    52     807F  8F    AB     03E9   471          BICW3   #^X807F, R2, R5         ; R5 = exponent(Y)
             54    55    A2     03EF   472          SUBW    R5, R4                  ; R4 = exponent(X) - exponent(Y)
          1D00 8F    54    B1     03F2   473          CMPW    R4, #58*128             ; compare R4 with 58
                   1B    14     03F7   474          BGTR    INF                     ; if X/Y > 2**57, branch to INF
                                03F9   475 ;
                                03F9   476 ; Test if Y > 0 or Y < 0
                                03F9   477 ;
                   52    B5     03F9   478          TSTW    R2                      ; test the sign of Y
                   14    14     03FB   479          BGTR    A2PLUS                  ; branch to A2PLUS if Y > 0
                   50    B5     03FD   480          TSTW    R0                      ; test the sign of X
                   08    18     03FF   481          BGEQ    A1PLUS                  ; branch to A1PLUS if X >= 0
                                0401   482 ;
                                0401   483 ; Y < 0 and X < 0 and X/Y =< 2**57
```

```
                                0401    484 ;
                          33    10    0401    485          BSBB    MTH$DATAN_R7D                   ; R0/R1 = DATAN(X/Y)
                50   FDB9 CF    62    0403    486          SUBD    D_PI, R0                        ; R0/R1 = -PI + DATAN(X/Y)
                                04    0408    487          RET                                     ; return
                                      0409    488 ;
                                      0409    489 ; Y < 0 and X > 0 and X/Y =< 2**57
                                      0409    490 ;
                                      0409    491 A1PLUS:
                          2B    10    0409    492          BSBB    MTH$DATAN_R7D                   ; R0/R1 = DATAN(X/Y)
                50   FDB1 CF    60    040B    493          ADDD    D_PI, R0                        ; R0/R1 = PI + DATAN(X/Y)
                                04    0410    494          RET                                     ; return
                                      0411    495 ;
                                      0411    496 ; Y > 0 and X/Y =< 2**57
                                      0411    497 ;
                                      0411    498 A2PLUS:
                          23    10    0411    499          BSBB    MTH$DATAN_R7D                   ; R0/R1 = DATAN(X/Y)
                                04    0413    500          RET                                     ; return
                                      0414    501 ;
                                      0414    502 ; Y = 0 or X/Y > 2**57
                                      0414    503 ;
                                      0414    504 INF:
                          50    B5    0414    505          TSTW    R0                              ; test the sign of X
                          08    14    0416    506          BGTR    1$                              ; branch if X > 0
                          0C    13    0418    507          BEQL    2$                              ; branch if X = 0
                50   FDB2 CF    70    041A    508          MOVD    D_MPI_OVER_2, R0                ; R0/R1 = DATAN(X/Y) = -PI/2
                                04    041F    509          RET                                     ; return
                                      0420    510
                50   FDA4 CF    70    0420    511 1$:      MOVD    D_PI_OVER_2, R0                 ; R0/R1 = DATAN(X/Y) = PI/2
                                04    0425    512          RET                                     ; return
                                      0426    513
                                      0426    514 ;+
                                      0426    515 ; Here if both X = 0 and Y = 0.  Signal INVALID ARG TO MATH LIBRARY
                                      0426    516 ;-
                                      0426    517
              50    01    0F    79    0426    518 2$:      ASHQ    #15, #1, R0                     ; R0/R1 = reserved operand, copied
                                      042A    519                                                 ; to CHF$L_MCH_SAVR0/R1 so handlers
                                      042A    520                                                 ; can change if they want to continue.
            7E    00'8F    9A    042A    521          MOVZBL  #MTH$K_INVARGMAT, -(SP)             ; code for INVALID ARG TO MATH LIBRARY
00000000'GF      01    FB    042E    522          CALLS   #1, G^MTH$$SIGNAL                       ; Signal SEVERE error
                                04    0435    523          RET                                     ; return if a handler says SS$_CONTINUE
```

```
                                0436    525          .SBTTL  MTH$DATAN_R7 - Special DATAN routine
                                0436    526
                                0436    527 ; Special DATAN - used by the standard routine, and directly.
                                0436    528 ;
                                0436    529 ; CALLING SEQUENCES:
                                0436    530 ;       save anything needed in R0:R7
                                0436    531 ;       MOVD    ...,R0                  ; input in R0/R1
                                0436    532 ;       JSB     MTH$DATAN_R7
                                0436    533 ;       return with result in R0/R1
                                0436    534 ; Note:  This  routine  is written to avoid causing any integer overflows,
                                0436    535 ;    floating overflows, or floating underflows or divide by 0 conditions,
                                0436    536 ;    whether enabled or not.
                                0436    537 ;
                                0436    538 ; REGISTERS USED:
                                0436    539 ;       R0/R1 - Floating argument then result
                                0436    540 ;       R0:R5 - POLYD
                                0436    541 ;       R6    - Pointer into DATAN_TABLE
                                0436    542 ;       R6/R7 - Y during POLYD
                                0436    543
                                0436    544
                                0436    545 MTH$DATAN_R7D:                          ; for local use only!
                     50   52    66  0436    546          DIVD    R2, R0
                                0439    547 MTH$DATAN_R7::                          ; Special DATAN routine
                          50    53  0439    548          TSTF    R0                 ; R6 = X = argument
                          76    19  043B    549          BLSS    NEG_ARG            ; Branch to negative argument logic
                                043D    550 ;
                                043D    551 ; Argument is positive
                                043D    552 ;
                56   50   3EC0 8F    A3  043D    553          SUBW3   #^X3EC0, R0, R6    ; Argument is less than 3/32,
                          47    19  0443    554          BLSS    SMALL              ;  branch to small argument logic
                     56   036F 8F    B1  0445    555          CMPW    #^X036F, R6        ; Argument is greater that 11,
                          43    19  044A    556          BLSS    LARGE_ARG          ;  branch to large argument logic
                                044C    557 ;
                                044C    558 ; Logic for positive medium sized arguments.  Get pointer into DATAN_TABLE.
                                044C    559 ;
                  56    56    FC 8F    9C  044C    560          ROTL    #-4, R6, R6        ; R6 = index into MTH$$AB_ATAN table
                56    FFFFFF00 8F    CA  0451    561          BICL    #-256, R6          ; zero high order bits of index
             56   00000000'GF46    90  0458    562          MOVB    G^MTH$$AB_ATAN[R6], R6 ; R6 = offset into DATAN_TABLE
                  56    FB9B CF46    7E  0460    563          MOVAQ   DATAN_TAB[E[R6], R6    ; R6 = pointer to XHI
                                0466    564 ;
                                0466    565 ; Compute Z
                                0466    566 ;
                     52   86    7D  0466    567          MOVQ    (R6)+, R2          ; R2 = XHI
                54   50   52    65  0469    568          MULD3   R2, R0, R4         ; R4 = X*XHI
                     54   08    60  046D    569          ADDD    #1, R4             ; R4 = 1 + X*XHI
                     50   52    62  0470    570          SUBD    R2, R0             ; R0 = X - XHI
                     50   54    66  0473    571          DIVD    R4, R0             ; R0 = Z = (X - XHI)/(1 + X*XHI)
                                0476    572 ;
                                0476    573 ; Evaluate Z*P(Z**2)
                                0476    574 ;
                     7E   50    7D  0476    575          MOVQ    R0, -(SP)          ; Push Z onto the stack
                     50   50    64  0479    576          MULD    R0, R0             ; R0 = Z**2
             FCCE CF      06    50  75  047C    577          POLYD   R0, #DATANLEN1-1, DATANTAB1
                                0482    578                                     ; R0 = P(Z**2)
                     50   8E    64  0482    579          MULD    (SP)+, R0          ; R0 = DATAN(Z) = Z*P(Z**2)
                     50   86    60  0485    580          ADDD    (R6)+, R0          ; R0 = DATAN_XHI_LO + DATAN(Z)
                     50   66    60  0488    581          ADDD    (R6), R0           ; R0 = DATAN(X) = DATAN_XHI_HI +
```

```
                                048B    582                                     ;       (DATAN_XHI_LO + DATAN(Z))
                             05 048B    583             RSB                     ; Return
                                048C    584
                                048C    585
                        0098    31 048C    586 SMALL:  BRW     SMALL_ARG           ; Dummy label used to avoid adding
                                048F    587                                     ;   an extra insrtuction in the
                                048F    588                                     ;   medium argument logic
                                048F    589 ;
                                048F    590 ; Large positive argument logic.
                                048F    591 ;
                                048F    592
                                048F    593 LARGE_ARG:
56   00000000 0000C080 8F    50 67 048F    594             DIVD3   R0, #-1, R6         ; R6 = -W = -1/X
                50    56    56 65 049B    595             MULD3   R6, R6, R0          ; R0 = W**2
             FCAB CF    06    50 75 049F    596             POLYD   R0, #DATANLEN1-1, DATANTAB1
                                04A5    597                                     ; R0 = P(W**2)
                50    56    64 04A5    598             MULD    R6, R0              ; R0 = DATAN(W) = -W*P(W**2)
                50    FD34  CF 60 04A8    599             ADDD    D_PI_OVER_2_LO, R0  ;
                50    FD27  CF 60 04AD    600             ADDD    D_PI_OVER_2_HI, R0  ; R0 = DATAN(X) = PI/2 - DATAN(W)
                             05 04B2    601             RSB                     ; Return
                                04B3    602
                                04B3    603 ;
                                04B3    604 ; Logic for negative arguments
                                04B3    605 ;
                                04B3    606
                                04B3    607 NEG_ARG:
             56    50    BEC0 8F    A3 04B3    608             SUBW3   #^XBEC0, R0, R6     ; Argument is less than 3/32,
                                6C 19 04B9    609             BLSS    SMALL_ARG           ;  branch to small argument logic
                56    036F 8F    B1 04BB    610             CMPW    #^X036F, R6         ; Argument is greater than 11,
                                41 19 04C0    611             BLSS    N_LARGE_ARG         ;  branch to large argument logic
                                04C2    612 ;
                                04C2    613 ; Logic for negative medium sized arguments.  Get index into DATAN_TABLE.
                                04C2    614 ;
          56    56    FC 8F    9C 04C2    615             ROTL    #-4, R6, R6         ; R6 = index into MTH$$AB_ATAN table
       56    FFFFFF00 8F    CA 04C7    616             BICL    #-256, R6           ; clear high order (unused) bits of ind
    56    00000000'GF46    90 04CE    617             MOVB    G^MTH$$AB_ATAN[R6], R6  ; R6 = offset into DATAN_TABLE
          56    FB25 CF46    7E 04D6    618             MOVAQ   DATAN_TAB[E[R6], R6 ; R6 = pointer to XHI
                                04DC    619 ;
                                04DC    620 ; Compute Z
                                04DC    621 ;
                   52    86    7D 04DC    622             MOVQ    (R6)+, R2           ; R2 = XHI
                54 50    52    65 04DF    623             MULD3   R2, R0, R4          ; R4 = X*XHI
                54 08    54    63 04E3    624             SUBD3   R4, #1, R4          ; R4 = 1 - X*XHI = 1 + X*(-XHI)
                50 52    60 04E7    625             ADDD    R2, R0              ; R0 = X + XHI = X - (-XHI)
                50 54    66 04EA    626             DIVD    R4, R0              ; R0 = Z
                                04ED    627 ;
                                04ED    628 ; Evaluate Z*P(Z**2)
                                04ED    629 ;
                   7E    50    7D 04ED    630             MOVQ    R0, -(SP)           ; Push Z onto the stack
                50 50    64 04F0    631             MULD    R0, R0              ; R0 = Z**2
FC57 CF            06    50 75 04F3    632             POLYD   R0, #DATANLEN1-1, DATANTAB1
                                04F9    633                                     ; R0 = P(Z**2)
                50    8E    64 04F9    634             MULD    (SP)+, R0           ; R0 = DATAN(Z) = Z*P(Z**2)
                50    86    62 04FC    635             SUBD    (R6)+, R0           ; R0 = DATAN_XHI_LO + DATAN(Z)
                50    66    62 04FF    636             SUBD    (R6), R0            ; R0 = DATAN(X) = DATAN_XHI_HI +
                                0502    637                                     ;       (DATAN_XHI_LO + DATAN(Z))
                             05 0502    638             RSB                     ; Return
```

```
                                0503    639 ;
                                0503    640 ; Logic for large negative  arguments
                                0503    641 ;
                                0503    642
                                0503    643 N_LARGE_ARG:
56   00000000 0000C080 8F    50   67    0503    644         DIVD3   R0, #-1, R6             ; R6 = W = 1/!X!
                50      56   56   65    050F    645         MULD3   R6, R6, R0              ; R0 = W**2
              FC37 CF   06   50   75    0513    646         POLYD   R0, #DATANLEN1-1, DATANTAB1
                                0519    647                                                 ; R0 = P(W**2)
                50      56   64   0519    648         MULD    R6, R0                  ; R0 = DATAN(W) = W*P(W**2)
              50   FCC0 CF   62   051C    649         SUBD    D_PI_OVER_2_LO, R0      ;
              50   FCB3 CF   62   0521    650         SUBD    D_PI_OVER_2_HI, R0      ; R0 = DATAN(X) = DATAN(W) - PI/2
                                05   0526    651         RSB                             ; Return
                                0527    652
                                0527    653 ;
                                0527    654 ; Small argument logic.
                                0527    655 ;
                                0527    656
                                0527    657 SMALL_ARG:
                50      56   7D   0527    658         MOVQ    R0, R6                  ; R6 = argument = X
              50   8000 8F   AA   052A    659         BICW    #^X8000, R0             ; R0 = !X!
              50   3280 8F   B1   052F    660         CMPW    #^X3280, R0             ; Compare 2^-28 to !X!
                           04   19   0534    661         BLSS    1$                      ; Branch to Polyinomial evaluation
                50      56   7D   0536    662         MOVQ    R6, R0                  ; Return with answer equal to argument
                           05   0539    663         RSB                             ;
                                053A    664
                50      50   64   053A    665 1$:     MULD    R0, R0                  ; R0 = X**2
              FC45 CF   06   50   75    053D    666         POLYD   R0, #DATANLEN2-1, DATANTAB2
                                0543    667                                                 ; R0 = Q(X**2)
                50      56   64   0543    668         MULD    R6, R0                  ; R0 = X*Q(X**2)
                50      56   60   0546    669         ADDD    R6, R0                  ; R0 = DATAN(X) = X + X*Q(X**2)
                           05   0549    670         RSB                             ; Return
                                054A    671
```

```
                     054A    673          .SBTTL  MTH$DATAND - Standard Single Precision Floating Arc Tangent
                     054A    674
                     054A    675
                     054A    676 ;++
                     054A    677 ; FUNCTIONAL DESCRIPTION:
                     054A    678 ;
                     054A    679 ; DATAND - double precision floating point function
                     054A    680 ;
                     054A    681 ; DATAN is computed using the following steps:
                     054A    682 ;
                     054A    683 ;    1. If X > 11 then
                     054A    684 ;       a. Let W = 1/X.
                     054A    685 ;       b. Compute DATAN(W) = W*P(W**2), where P is a polynomial of
                     054A    686 ;       degree 6.
                     054A    687 ;       c. Set DATAN(X) = pi/2 - DATAN(W)
                     054A    688 ;    2. If 3/32 =< X =< 11 then
                     054A    689 ;       a. Obtain XHI by table look-up.
                     054A    690 ;       b. Compute Z = (X - XHI)/(1 + X*XHI).
                     054A    691 ;       c. Compute DATAN(Z) = Z*P(Z**2), where P is a polynomial of
                     054A    692 ;       degree 6.
                     054A    693 ;       d. Obtain DATAN(XHI) by table look-up.  DATAN(XHI) will have
                     054A    694 ;       two parts - the high order bits, DATAN_XHI_HI, and the low
                     054A    695 ;       order bits, DATAN_XHI_LO.
                     054A    696 ;       e. Compute DATAN(X) = DATAN_XHI_HI + (DATAN_XHI_LO + DATAN(Z)).
                     054A    697 ;    3. If 0 =< X < 3/32 then
                     054A    698 ;       a. Compute DATAN(X) = X + X*Q(X**2), where Q is a polynomial
                     054A    699 ;       of degree 6.
                     054A    700 ;    4. If X < 0 then
                     054A    701 ;       a. Compute Y = DATAN(|X|) using steps 1 to 3.
                     054A    702 ;       b. Set DATAN(X) = -Y.
                     054A    703 ;
                     054A    704 ; CALLING SEQUENCE:
                     054A    705 ;
                     054A    706 ;       Arctangent.wd.v = MTH$DATAND(x.rd.r)
                     054A    707 ;
                     054A    708 ; INPUT PARAMETERS:
                     054A    709
00000004             054A    710          LONG = 4                        ; define longword multiplier
00000004             054A    711          x = 1 * LONG                    ; x is an angle in radians
                     054A    712 ;
                     054A    713 ; IMPLICIT INPUTS:      none
                     054A    714 ;
                     054A    715 ; OUTPUT PARAMETERS:
                     054A    716 ;
                     054A    717 ;       VALUE:  double precision floating arctangent angle of the argument
                     054A    718 ;
                     054A    719 ; IMPLICIT OUTPUTS:     none
                     054A    720 ;
                     054A    721 ; SIDE EFFECTS:
                     054A    722 ;
                     054A    723 ; Signals:      none
                     054A    724 ;
                     054A    725 ; NOTE: This  procedure disables floating point underflow, enable integer
                     054A    726 ;    overflow, causes no floating overflow or other arithmetic traps, and
                     054A    727 ;    preserves enables across the call.
                     054A    728 ;
                     054A    729 ;--
```

```
                                054A   730
                                054A   731
                        40FC    054A   732          .ENTRY  MTH$DATAND, ACMASK      ; standard call-by-reference entry
                                054C   733                                         ; disable DV (and FU), enable IV
                                054C   734          MTH$FLAG_JACKET                ; flag that this is a jacket procedure
                                054C
    6D  00000000'GF     9E      054C               MOVAB   G^MTH$$JACKET_HND, (FP)
                                0553                                            ; set handler address to jacket
                                0553                                            ; handler
                                0553
                                0553   735                                      ; in case of an error in special JSB
                                0553   736                                      ; routine
    50  04 BC          70      0553   737          MOVD    @x(AP), R0             ; R0/R1 = arg
                    6A  10      0557   738          BSBB    MTH$DATAND_R7          ; call special DATAND rountine
                        04      0559   739          RET                            ; return - result in R0
                                055A   740
```

```
                     055A   742          .SBTTL  MTH$DATAND2 - Standard Double Floating Arctangent With 2 Arguments
                     055A   743 ;++
                     055A   744 ; FUNCTIONAL DESCRIPTION:
                     055A   745 ;
                     055A   746 ; DATAND2  - double precision floating point function
                     055A   747 ;
                     055A   748 ; DATAND2(X,Y) is computed as following:
                     055A   749 ;
                     055A   750 ;       If Y = 0 or X/Y > 2**57, DATAND2(X,Y) = 90 * (sign X)
                     055A   751 ;       If Y > 0 and X/Y =< 2**57, DATAND2(X,Y) = DATAND(X/Y)
                     055A   752 ;       If Y < 0 and X/Y =< 2**57, DATAND2(X,Y) = 180 * (sign X) + DATAND(X/Y)
                     055A   753 ;
                     055A   754 ;
                     055A   755 ; CALLING SEQUENCE:
                     055A   756 ;
                     055A   757 ;       Arctangent2.wd.v = MTH$DATAND2(x.rd.r, y.rd.r)
                     055A   758 ;
                     055A   759 ; INPUT PARAMETERS:
                     055A   760
00000004             055A   761 ;       x = 1 * LONG                    ; x is the first argument
00000008             055A   762 ;       y = 2 * LONG                    ; y is the second argument
                     055A   763 ;
                     055A   764 ; SIDE EFFECTS: See description of MTH$DATAND
                     055A   765 ;
                     055A   766 ;--
                     055A   767
                     055A   768
                 40FC 055A   769          .ENTRY  MTH$DATAND2, ACMASK     ; standard call-by-reference entry
                     055C   770                                          ; disable DV (and FU), enable IV
                     055C   771          MTH$FLAG_JACKET                 ; flag that this is a jacket procedure
                     055C   772
6D   00000000'GF     9E  055C            MOVAB   G^MTH$$JACKET_HND, (FP)
                     0563                                                ; set handler address to jacket
                     0563                                                ; handler
                     0563   773
                     0563   774                                          ; in case of an error in special JSB
                     0563   775                                          ; routine
     50   04 BC     70  0563   776          MOVD    @x(AP), R0              ; R0/R1 = arg1
     52   08 BC     70  0567   777          MOVD    @y(AP), R2              ; R2/R3 = arg2
                     056B   778 ;
                     056B   779 ; Test if Y = 0 or X/Y > 2**57
                     056B   780 ;
                 31  13  056B   781          BEQL    INF_DEG                 ; branch to INF_DEG if Y = 0
54   50   807F 8F    AB  056D   782          BICW3   #^X807F, R0, R4         ; R4 = exponent(X)
55   52   807F 8F    AB  0573   783          BICW3   #^X807F, R2, R5         ; R5 = exponent(Y)
          54   55    A2  0579   784          SUBW    R5, R4                  ; R4 = exponent(X) - exponent(Y)
     1D00 8F   54    B1  057C   785          CMPW    R4, #58*128             ; compare R4 with 58
                 1B  14  0581   786          BGTR    INF_DEG                 ; if X/Y > 2**57, branch to INF_DEG
                     0583   787 ;
                     0583   788 ; Test if Y > 0 or Y < 0
                     0583   789 ;
                 52  B5  0583   790          TSTW    R2                      ; test the sign of Y
                 14  14  0585   791          BGTR    A2PLUSD                 ; branch to A2PLUSD if Y > 0
                 50  B5  0587   792          TSTW    R0                      ; test the sign of X
                 08  18  0589   793          BGEQ    A1PLUSD                 ; branch to A1PLUSD if X >= 0
                     058B   794 ;
                     058B   795 ; Y < 0 and X < 0 and X/Y =< 2**57
```

```
                                0588   794 ;
                          33    10 0588   795          BSBB    MTH$DATAND_R7D          ; R0/R1 = DATAND(X/Y)
                50  FE27    CF    62 058D   796          SUBD    D_180, R0               ; R0/R1 = -180 + DATAND(X/Y)
                                04 0592   797          RET                             ; return
                                   0593   798 ;
                                   0593   799 ; Y < 0 and X > 0 and X/Y =< 2**57
                                   0593   800 ;
                                   0593   801 A1PLUSD:
                          2B    10 0593   802          BSBB    MTH$DATAND_R7D          ; R0/R1 = DATAND(X/Y)
                50  FE1F    CF    60 0595   803          ADDD    D_180, R0               ; R0/R1 = 180 + DATAND(X/Y)
                                04 059A   804          RET                             ; return
                                   059B   805 ;
                                   059B   806 ; Y > 0 and X/Y =< 2**57
                                   059B   807 ;
                                   059B   808 A2PLUSD:
                          23    10 059B   809          BSBB    MTH$DATAND_R7D          ; R0/R1 = DATAND(X/Y)
                                04 059D   810          RET                             ; return
                                   059E   811 ;
                                   059E   812 ; Y = 0 or X/Y > 2**57
                                   059E   813 ;
                                   059E   814 INF_DEG:
                          50    B5 059E   815          TSTW    R0                      ; test the sign of X
                          08    14 05A0   816          BGTR    1$                      ; branch if X > 0
                          0C    13 05A2   817          BEQL    2$                      ; branch if X = 0
                50  FE08    CF    70 05A4   818          MOVD    D_M90, R0       ; R0/R1 = DATAND(X/Y) = -90
                                04 05A9   819          RET                             ; return
                                   05AA   820
                50  FDFA    CF    70 05AA   821 1$:      MOVD    D_90, R0                ; R0/R1 = DATAND(X/Y) = 90
                                04 05AF   822          RET                             ; return
                                   05B0   823
                                   05B0   824 ;+
                                   05B0   825 ; Here if both X = 0 and Y = 0.  Signal INVALID ARG TO MATH LIBRARY
                                   05B0   826 ;-
                                   05B0   827
              50    01    0F    79 05B0   828 2$:      ASHQ    #15, #1, R0             ; R0/R1 = reserved operand, co180ed
                                   05B4   829                                          ; to CHF$L_MCH_SAVR0/R1 so handlers
                                   05B4   830                                          ; can change if they want to continue.
                    7E  00'8F    9A 05B4   831          MOVZBL  #MTH$K_INVARGMAT, -(SP) ; code for INVALID ARG TO MATH LIBRARY
00000000'GF         01    FB 05B8   832          CALLS   #1, G^MTH$$SIGNAL               ; Signal SEVERE error
                                04 05BF   833          RET                             ; return if a handler says SS$_CONTINUE
```

```
                                                B 1
                                05C0    835          .SBTTL  MTH$DATAND_R7 - Special DATAND routine
                                05C0    836
                                05C0    837 ; Special DATAND - used by the standard routine, and directly.
                                05C0    838 ;
                                05C0    839 ; CALLING SEQUENCES:
                                05C0    840 ;       save anything needed in R0:R7
                                05C0    841 ;       MOVD    ...,R0                  ; input in R0/R1
                                05C0    842 ;       JSB     MTH$DATAND_R7
                                05C0    843 ;       return with result in R0/R1
                                05C0    844 ; Note: This routine is written to avoid causing any  integer  overflows,
                                05C0    845 ;   floating overflows, or floating underflows or divide by 0 conditions,
                                05C0    846 ;   whether enabled or not.
                                05C0    847 ;
                                05C0    848 ; REGISTERS USED:
                                05C0    849 ;       R0/R1 - Floating argument then result
                                05C0    850 ;       R0:R5 - POLYD
                                05C0    851 ;       R6    - Pointer into DATAND_TABLE
                                05C0    852 ;       R6/R7 - Y during POLYD
                                05C0    853
                                05C0    854
                                05C0    855 MTH$DATAND_R7D:                         ; for local use only!
                     50  52  66  05C0    856          DIVD    R2, R0
                                05C3    857 MTH$DATAND_R7::                         ; Special DATAND routine
                         50  53  05C3    858          TSTF    R0                     ; R6 = X = argument
                         71  19  05C5    859          BLSS    NEG_ARGD               ; Branch to negative argument logic
                                05C7    860 ;
                                05C7    861 ; Argument is positive
                                05C7    862 ;
              56  50  3EC0 8F   A3  05C7    863          SUBW3   #^X3EC0, R0, R6        ; Argument is less than 3/32,
                         47  19  05CD    864          BLSS    SMALLD                 ;  branch to small argument logic
                  56  036F 8F   B1  05CF    865          CMPW    #^X036F, R6            ; Argument is greater that 11,
                         43  19  05D4    866          BLSS    LARGE_ARGD             ;  branch to large argument logic
                                05D6    867 ;
                                05D6    868 ; Logic for positive medium sized arguments.  Get pointer into DATAND_TABLE.
                                05D6    869 ;
                56  56    FC 8F   9C  05D6    870          ROTL    #-4, R6, R6            ; R6 = index into AB_ATAN table
              56  FFFFFF00 8F   CA  05DB    871          BICL    #-256, R6              ; zero high order bits of index
           56  00000000'GF46   90  05E2    872          MOVB    G^MTH$$AB_ATAN[R6], R6 ; R6 = offset into DATAND_TABLE
                56  FBF9 CF46   7E  05EA    873          MOVAQ   DATAND_TABLE[R6], R6   ; R6 = pointer to XHI
                                05F0    874 ;
                                05F0    875 ; Compute z
                                05F0    876 ;
                     52  86   7D  05F0    877          MOVQ    (R6)+, R2              ; R2 = XHI
                54   50  52   65  05F3    878          MULD3   R2, R0, R4             ; R4 = X*XHI
                     54  08   60  05F7    879          ADDD    #1, R4                 ; R4 = 1 + X*XHI
                     50  52   62  05FA    880          SUBD    R2, R0                 ; R0 = X - XHI
                     50  54   66  05FD    881          DIVD    R4, R0                 ; R0 = Z = (X - XHI)/(1 + X*XHI)
                                0600    882 ;
                                0600    883 ; Evaluate Z*P(Z**2)
                                0600    884 ;
                     7E  50   7D  0600    885          MOVQ    R0, -(SP)              ; Push Z onto the stack
                     50  50   64  0603    886          MULD    R0, R0                 ; R0 = Z**2
          FD2C CF    06  50   75  0606    887          POLYD   R0, #DATANDLEN1-1, DATANDTAB1
                                060C    888                                         ; R0 = P(Z**2)
                     50  8E   64  060C    889          MULD    (SP)+, R0              ; R0 = DATAND(Z) = Z*Q(Z**2)
                     50  86   60  060F    890          ADDD    (R6)+, R0              ; R0 = DATAND_XHI_LO + DATAND(Z)
                     50  66   60  0612    891          ADDD    (R6), R0               ; R0 = DATAND(X) = DATAND_XHI_HI +
```

```
                                0615   892                                     ;       (DATAND_XHI_LO + DATAND(Z))
                          05    0615   893             RSB                     ; Return
                                0616   894
                                0616   895
                     008E   31  0616   896 SMALLD: BRW     SMALL_ARGD          ; Dummy label used to avoid adding
                                0619   897                                     ;   an extra insrtuction in the
                                0619   898                                     ;   medium argument logic
                                0619   899 ;
                                0619   900 ; Large positive argument logic.
                                0619   901 ;
                                0619   902
                                0619   903 LARGE_ARGD:
56  00000000 0000C080 8F  50    67  0619   904             DIVD3   R0, #-1, R6          ; R6 = -W = -1/X
            50   56    56    65  0625   905             MULD3   R6, R6, R0          ; R0 = W**2
          FD09 CF    06    50   75  0629   906             POLYD   R0, #DATANDLEN1-1, DATANDTAB1
                                062F   907                                     ; R0 = P(W**2)
               50    56    64   062F   908             MULD    R6, R0              ; R0 = -DATAND(Z) = -Z*P(W**2)
             50    FD72 CF    60  0632   909             ADDD    D_90, R0            ; R0 = DATAND(X) = 90 - DATAND(Z)
                          05    0637   910             RSB                     ; Return
                                0638   911
                                0638   912 ;
                                0638   913 ; Logic for negative arguments
                                0638   914 ;
                                0638   915
                                0638   916 NEG_ARGD:
          56    50    BEC0 8F   A3  0638   917             SUBW3   #^XBEC0, R0, R6     ; Argument is less than 3/32,
                          67    19  063E   918             BLSS    SMALL_ARGD          ;  branch to small argument logic
               56   036F 8F   B1  0640   919             CMPW    #^X036F, R6         ; Argument is greater than 11,
                          41    19  0645   920             BLSS    N_LARGE_ARGD        ;  branch to large argument logic
                                0647   921 ;
                                0647   922 ; Logic for negative medium sized arguments.  Get index into DATAND_TABLE.
                                0647   923 ;
  56    56     FC 8F    9C  0647   924             ROTL    #-4, R6, R6         ; R6 = index into MTH$$AB_ATAN table
 56     FFFFFF00 8F    CA  064C   925             BICL    #-256, R6           ; clear high order (unused) bits of ind
56    00000000'GF46    90  0653   926             MOVB    G^MTH$$AB_ATAN[R6], R6  ; R6 = offset into DATAN_TABLE
   56    FB88 CF46    7E  065B   927             MOVAQ   DATAND_TABLE[R6], R6    ; R6 = pointer to XHI
                                0661   928 ;
                                0661   929 ; Compute Z
                                0661   930 ;
          52    86    7D  0661   931             MOVQ    (R6)+, R2           ; R2 = XHI
       54 50    52    65  0664   932             MULD3   R2, R0, R4          ; R4 = X*XHI
       54 08    54    63  0668   933             SUBD3   R4, #1, R4          ; R4 = 1 - X*XHI = 1 + X*(-XHI)
          50    52    60  066C   934             ADDD    R2, R0              ; R0 = X + XHI = X - (-XHI)
          50    54    66  066F   935             DIVD    R4, R0              ; R0 = Z
                                0672   936 ;
                                0672   937 ; Evaluate Z*P(Z**2)
                                0672   938 ;
          7E    50    7D  0672   939             MOVQ    R0, -(SP)           ; Push Z onto the stack
          50    50    64  0675   940             MULD    R0, R0              ; R0 = Z**2
FCBA CF   06    50    75  0678   941             POLYD   R0, #DATANDLEN1-1, DATANDTAB1
                                067E   942                                     ; R0 = P(Z**2)
          50    8E    64  067E   943             MULD    (SP)+, R0           ; R0 = DATAND(Z) = Z*P(Z**2)
          50    86    62  0681   944             SUBD    (R6)+, R0           ; R0 = DATAND_XHI_LO + DATAND(Z)
          50    66    62  0684   945             SUBD    (R6), R0            ; R0 = DATAND(X) = DATAND_XHI_HI +
                                0687   946                                     ;       (DATAND_XHI_LO + DATAND(Z))
                          05    0687   947             RSB                     ; Return
                                0688   948 ;
```

```
                                0688   949 ; Logic for large negative  arguments
                                0688   950 ;
                                0688   951
                                0688   952 N_LARGE_ARGD:
56   00000000 0000C080 8F    50  67  0688   953          DIVD3   R0, #-1, R6              ; R6 = W = 1/!X!
                 50    56    56  65  0694   954          MULD3   R6, R6, R0              ; R0 = W**2
                    FC9A CF    06  50  75  0698   955          POLYD   R0, #DATANDLEN1-1, DATANDTAB1
                                069E   956                                              ; R0 = P(W**2)
                         50    56  64  069E   957          MULD    R6, R0              ; R0 = DATAND(W) = W*P(W**2)
                              50   FD03 CF  62  06A1   958          SUBD    D_90, R0            ; R0 = DATAND(X) = DATAND(W) - 90
                                   05  06A6   959          RSB                         ; Return
                                06A7   960
                                06A7   961 ;
                                06A7   962 ; Small argument logic.
                                06A7   963 ;
                                06A7   964
                                06A7   965 SMALL_ARGD:
                         56    50  70  06A7   966          MOVD    R0, R6              ; R6 = argument = X
                                   28  13  06AA   967          BEQL    3$                  ;
                              50   8000 8F  AA  06AC   968          BICW    #^X8000, R0         ; R0 = !X!
                              50   3280 8F  B1  06B1   969          CMPW    #^X3280, R0         ; Compare 2^-28 to !X!
                                   08  19  06B6   970          BLSS    1$                  ; Branch to Polynomial evaluation
                    50   56    FCE4 CF  65  06B8   971          MULD3   D_PI_OV_180_M_64, R6 ,R0; R0 = X*(pi/180 - 64)
                                   0C  11  06BE   972          BRB 2$                      ;
                         50    50  64  06C0   973 1$:       MULD    R0, R0              ; R0 = X**2
                    FCA7 CF    06  50  75  06C3   974          POLYD   R0, #DATANDLEN2-1, DATANDTAB2
                                06C9   975                                              ; R0 = Q(X**2)
                         50    56  64  06C9   976          MULD    R6, R0              ; R0 = X*Q(X**2)
                              56   0300 8F  A0  06CC   977 2$:       ADDW    #^X300, R6          ; R6 = X*2**6
                                50   56  60  06D1   978          ADDD    R6, R0              ; R0 = DATAND(X) = X*2**6 + X*Q(X**2)
                                   05  06D4   979 3$:       RSB                         ; Return
                                06D5   980
                                06D5   981
                                06D5   982          .END
```

```
A1PLUS                          00000409 R      01
A1PLUSD                         00000593 R      01
A2PLUS                          00000411 R      01
A2PLUSD                         0000059B R      01
ACMASK                        = 000040FC
DATANDLEN1                    = 00000007
DATANDLEN2                    = 00000007
DATANDTAB1                      00000338 R      01
DATANDTAB2                      00000370 R      01
DATAND_TABLE                    000001E8 R      01
DATANLEN1                     = 00000007
DATANLEN2                     = 00000007
DATANTAB1                       00000150 R      01
DATANTAB2                       00000188 R      01
DATAN_TABLE                     00000000 R      01
D_180                           000003B8 R      01
D_90                            000003A8 R      01
D_M90                           000003B0 R      01
D_MPI_OVER_2                    000001D0 R      01
D_PI                            000001C0 R      01
D_PI_OVER_2                     000001C8 R      01
D_PI_OVER_2_HI                  000001D8 R      01
D_PI_OVER_2_LO                  000001E0 R      01
D_PI_OV_180_M_64                000003A0 R      01
INF                             00000414 R      01
INF_DEG                         0000059E R      01
LARGE_ARG                       0000048F R      01
LARGE_ARGD                      00000619 R      01
LONG                          = 00000004
MTH$$AB_ATAN                    ********   X    00
MTH$$JACKET_HND                 ********   X    01
MTH$$SIGNAL                     ********   X    00
MTH$DATAN                       000003C0 RG     01
MTH$DATAN2                      000003D0 RG     01
MTH$DATAND                      0000054A RG     01
MTH$DATAND2                     0000055A RG     01
MTH$DATAND_R7                   000005C3 RG     01
MTH$DATAND_R7D                  000005C0 R      01
MTH$DATAN_R7                    00000439 RG     01
MTH$DATAN_R7D                   00000436 R      01
MTH$K_INVARGMAT                 ********     X  00
NEG_ARG                         000004B3 R      01
NEG_ARGD                        00000638 R      01
N_LARGE_ARG                     00000503 R      01
N_LARGE_ARGD                    00000688 R      01
SMALL                           0000048C R      01
SMALLD                          00000616 R      01
SMALL_ARG                       00000527 R      01
SMALL_ARGD                      000006A7 R      01
X                             = 00000004
Y                             = 00000008
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
_MTH$CODE	000006D5 (1749.)	01 (1.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.10	00:00:01.09
Command processing	152	00:00:00.70	00:00:03.81
Pass 1	115	00:00:02.50	00:00:07.34
Symbol table sort	0	00:00:00.02	00:00:00.20
Pass 2	178	00:00:02.11	00:00:06.41
Symbol table output	8	00:00:00.05	00:00:00.08
Psect synopsis output	2	00:00:00.02	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	487	00:00:05.51	00:00:19.18

The working set limit was 1050 pages.
16195 bytes (32 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 51 non-local and 8 local symbols.
1042 source lines were read in Pass 1, producing 22 object records in Pass 2.
1 page of virtual memory was used to define 1 macro.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB;2	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

```
MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:MTHDATAN/OBJ=OBJ$:MTHDATAN MSRC$:MTHJACKET/UPDATE=(ENH$:MTHJACKET)+MSRC
```

MTHCVTDG
LIS

MTHDACOS
LIS

MTHCGABS
LIS

MTHCDSINC
LIS

MTHCLOG
LIS

MTHDASIN
LIS

MTHCGLOG
LIS

MTHCONVER
LIS

MTHDATAN
LIS

MTHCSQRT
LIS

MTHCEXP
LIS

MTHCGSQRT
LIS

MTHCGEXP
LIS

MTHCSINCO
LIS

MTHCONJG
LIS

MTHCDSQRT
LIS

MTHCGSINC
LIS

MTHCOSH
LIS